Design and Implementation of a Menu Based OSCAR Command Line Interface

Wesley Bland[1,2], Thomas Naughton[1], Geoffroy Vallée[1], and Stephen L. Scott[1] *

[1]Oak Ridge National Laboratory
Computer Science and Mathematics Division
Oak Ridge, TN 37831, USA

[2]Tennessee Technological University
Cookeville, TN 38505, USA

Abstract

The Open Source Cluster Application Resources (OSCAR) toolkit is used to build and maintain HPC clusters. The OSCAR cluster installer provides a graphical user interface (GUI) "wizard" that directs the user through the installation process. This GUI is useful for general usage, but in some instances a more basic interface is desirable. Therefore, a command line interface (CLI) has been developed for the OSCAR installer. Through the CLI, the OSCAR installer is better equipped to support scripting, which aids more advanced testing scenarios and helps in cluster replication. Additionally, the CLI is better suited for remote installations using OSCAR.

While a graphical user interface simplifies installation for occasional users, frequent users and the OSCAR developers will find the CLI beneficial. For developers, the CLI will make testing easier because it may now be scripted for automated testing. Furthermore, because the CLI creates logs of user input, the user may create an exact duplicate of a previous installation, which could be used in later rebuilds or for diagnostic purposes when reporting issues to the developers. This paper provides an overview of the design and capabilities of the OSCAR CLI along with a brief discussion of possible future work.

1 Introduction

The Open Source Cluster Application Resources (OSCAR) toolkit is used to build and maintain HPC clusters [6]. Since its release in April 2001, the OSCAR cluster installer has offered a graphical user interface (GUI) that directs the user through the installation process. This GUI "OSCAR Wizard" is useful for general users, but for quite some time there has been interest in the OSCAR community for a more basic interface to manage the installation process. In response, a command line interface (CLI) to OSCAR has been developed.

The CLI expands the capabilities of OSCAR, allowing for improved automation and replication. Additionally, the CLI is better suited for lower bandwidth remote installations where the graphical X Window environment was often too cumbersome. The CLI can be used by administrators (and developers) during cluster testing by providing a more "script friendly" interface to aid automation.

The remainder of the paper includes, a brief overview of past work (Section 2) as well as the motivation (Section 3) and design (Section 4) for the OSCAR CLI. Followed by a description of the current CLI implementation (Section 5) along with some comments for future work (Section 6) and concluding remarks (Section 7).

2 Background

OSCAR bundles tools for building, programming, and using High Performance Computing (HPC) clusters [6]. It includes many software applications combined to form "OSCAR Packages" that give basic and advanced functionality for clusters. There are also tools for cluster installation and maintenance.

A basic tenet of OSCAR is to leverage previously accepted solutions whenever possible to avoid duplication of effort. Thus, the "core" (required) infrastructure includes well known tools for cluster administration. These include: System Installation Suite (SIS) [1], Cluster, Command, and

*ORNL's work was supported by the U.S. Department of Energy, under Contract DE-AC05-00OR22725.

[1]SIS is comprised of SystemInstaller, SystemImager and SystemConfigurator. All tools are generally available but the SystemInstaller and Sys-

Control (C3), Env-Switcher/Modules, and base OSCAR libraries/scripts [2, 3, 4, 5].

Previously, des Ligneris and Camargos [1] outlined a plan for an OSCAR CLI. This paper discussed a method where all inputs for the OSCAR install were derived from the command line as opposed to the standard graphical OSCAR Wizard. The approached described separate command line tools that mirrored much of the graphical interface, providing a rich interface available entirely from a text-only console. However, this work was never introduced into the central OSCAR development repository. Additionally, the approach faced some disadvantages that the current approach seeks to overcome. These include a more complex usage scenarios due to the increased flexibility of separate CLI commands and no simple way to force sequencing to accommodate the ordering expected by OSCAR. This latter aspect, while possible under the multiple command approach, introduces a risk where a user could accidently execute steps in an order that would cause problems or errors during the installation.

There were also more elaborate designed plans to overhaul the OSCAR interface entirely. The idea being to change OSCAR to follow a state-machine approach with menus produced for each stage of the install. The basic premise being similar to that of standard Linux distribution install engines where early steps may use an alternate interface, e.g., `ncurses`, before setting up the X Window system and then using an X based interface. The preliminary design and initial coding was developed under the project name, Meta Menu [8]. However, project objectives changed as did developer time allocations, leaving the project orphaned before completion.

3 Motivation

A key advantage to a GUI is that it allows inexperienced users to interact with the software tool. This is achieved by presenting the software's capabilities through standard user interface idioms, e.g., buttons for available actions (commands). The advantage being that a user does not have to memorize a list of commands in order to use the tool. Conversely, if the interface is strictly CLI based the learning curve can be much steeper, where all commands must be known by the user in order to make use of the software [7]. However, one approach to overcome the overhead associated with a CLI is to use menus. In this scenario, the user is able to select actions offered by the tool, which offers advantages similar to those of a GUI albeit in a much more basic format. Lastly, a CLI has the advantage of being more conducive to automation because the input/output is text-based.

temConfigurator components are the most commonly used pieces outside of OSCAR.

The OSCAR toolkit has employed a GUI to assist in the installation and management of HPC clusters with a minimum amount of user experience. This OSCAR GUI "Wizard" has been sufficient thus far in the project and fit well with the general project goals of user simplicity. The GUI gives the user quick access to OSCAR without requiring a complex understanding of its inner workings. The OSCAR GUI accomplishes this by leading the user through the installation and hiding extra details and capabilities. However, in some instances an alternate interface would be beneficial to some advanced users and OSCAR developers.

For example, as the number of supported Linux distributions and hardware architectures grow the need for a simpler user interface becomes more important. Every time a new distribution is added to OSCAR, it must be tested extensively to ensure that it works well with the existing OSCAR code. The addition of a command line interface (CLI) increases the options users have to interact with OSCAR and creates a greater potential for automation. This latter aspect becomes paramount as the supported platforms grow in number and testing and validation of OSCAR increases in complexity.

As mentioned in Section 2, there have been previous attempts to develop an alternative to the current OSCAR user interface. The prior work [1] was entirely command line oriented and Meta Menu [8] sought to provide a more versatile approach that would allow for different presentation layers, e.g., graphical, text-based. The current approach tries to take an intermediate approach, where the usability of the graphical environment is reflected in a menu-based interface, and the power of the command line environment comes from the ability to automate and reuse installations.

In Spolsky's discussion of user interfaces [7], he cites the menu-based approach to designing an interface as a way to improve usability. While the GUI has a more user-friendly design with a click-able interface, the text-based installation should still have some measure of interactive convenience. The menu-based interface used in the text-based version is described in detail in Section 4. Both the past and current CLI approaches offer automation via scripting, but the current approach actually saves the user input as it is entered for later reuse. Otherwise the user is forced to know all commands and options a priori. Additionally, the separation of the commands causes (potential) problems if the proper command sequence is not maintained, based on the rules of ordering OSCAR currently expects. The current text-only menu based approach, that saves input for later reuse, honors both the ordering requirements, and enables automation while maintaining a reasonable usage model.

One phase of development that is under constant improvement is *testing*. It is one of the most important stages where many bugs could be caught and corrected, but in order to catch these bugs, good testing practices must be in place. The OSCAR CLI has been written to be one of these good practices. By using the CLI, developers can automate the testing process which both speeds it up and improves its accuracy. Testing can be completed faster when automated, and is more accurate without the need for human interaction.

Another way the OSCAR CLI improves the OSCAR project is to allow scripting. Many users do not want to have to manually perform the same steps every time they do an installation and would like some way to do this automatically. The new OSCAR CLI logs provide a way to do this by writing to files the input the user gives during a CLI installation. As the user types, the input is stored in a log that can later be fed back into the installer as an input file. Without any modification, an entire installation can be replicated. Any or all of the steps can be done this way allowing the user to customize the installation without having to be a part of the process. Also, the user can modify the scripts to do anything that can be done with the interactive installation. For example, if a user wants to add another node to the cluster, he can change the logs from the *Define OSCAR Clients* step to include one more node and without changing anything else, can have the same installation as before with one more node.

One thing users want is to reduce the overhead on their machines as much as possible. The X Windows System is a process that introduces the most overhead on a Linux machine. For many users, the only time they use the graphical interface on their cluster is when they are installing OSCAR. Without the graphical OSCAR interface, the administrator would not need to run an X server on their machine and would save resources. This also allows the OSCAR installer to be run over network lines that do not have enough bandwidth to run a graphical interface. Many cluster users do not always sit in front of the machines they are using and being able to run the OSCAR installer remotely would be very helpful. In order to provide both interfaces (graphical and command line), the graphical libraries must still be installed on the head node, but the server does not need to be running. If at a later time the administrator wants to use the graphical version of the installation, the tools will still be there and all that will be necessary is to start the X server.

Including the new CLI interface, OSCAR now has two completely different types of interfaces for users. However, many users might find another way that would work better for them, or perhaps want to combine the use of OSCAR with other software. Using the CLI, the interface for OSCAR has become much more extensible to allow other methods of interaction with users. Any other kind of interface can sit on top of the CLI and make system calls while getting user input in the way that most appropriately helps the user. Another way the CLI could be leveraged would be if vendors were to take advantage of the new tool. Using the CLI, they could write their own version of the OSCAR installation that is pre-customized for their own clusters. This tool could be shipped out with the machines, providing the buyers with a simple way to set up their new cluster.

4 Design

The CLI was designed to be as close to a mirror of the GUI as possible so that a user could pick an interface not based on functionality, but by which one would be most useful. For the most part this goal has been achieved. Most steps in the CLI closely mirror the GUI and will seem identical to the user. For a few steps there have been some minor changes to the interface, but all the functionality provided in the GUI is still present in the CLI. With one exception, the optional GUI based *Configurator* is not accessible from the CLI (see Section 5 for details). The *Configurator* is the part of the OSCAR installation that allows the user to change any default configurations in the packages installed with OSCAR.

One thing the GUI does a very good job of is to make sure that the user does not get ahead of himself when completing the OSCAR installation. This is a feature that has been maintained in the CLI. The CLI guides the user through the steps of the OSCAR installation process making sure that each part is completed before moving on to the next one. However, if the user does want to skip a step, it is possible to do so by using command line flags. By maintaining this progressive installation, the extensibility of the CLI is maintained for other interfaces. The other interface (or script) can communicate with the CLI using files with the commands pre-loaded into them. These files have all the same responses that a user would give if going through a normal installation using the interactive form. In fact, as a user completes the installation, these files are automatically generated and saved for easy reproducibility later if the user will be reinstalling the cluster.

Before the final step in the OSCAR installation, the nodes must be rebooted individually so they can receive their disk images from the server. This is the only step that cannot be automated. Each different kind of cluster has its own way to reboot the nodes and load the new images. For example, with real physical nodes, the computers must be physically rebooted with a way to load the images (i.e. PXE). However, if the cluster is a virtual cluster or a combination of the two, the new nodes may simply need to be started up with a console command. To handle this problem the command line version of OSCAR allows the user to let the program know when to continue on in the installation.

5 Implementation

The new CLI was implemented using Perl to maintain consistency with the existing OSCAR code base. For the most part, the new CLI scripts are fairly well isolated from the other OSCAR code and use the existing libraries to interact with the components of OSCAR. The exception to this is the *Setup Networking* step in the GUI. In this area, the code is very tightly coupled and much of the logic is embedded in the GUI as opposed to a supporting library. This resulted in CLI specific code being added to the same module used by the GUI. Additionally, the tight coupling led to code duplication. More details of this problem are discussed in Section 6

Six of the eight steps in the OSCAR installation are direct translations of the GUI interface to a text-based mode. These steps should directly mirror the GUI mode that most users are used to. The *Setup Network* step is slightly different than the GUI because of an issue with the MAC address selection. In the GUI, MAC addresses can either be: (i) read from a file, or (ii) automatically detected when the machines bootstrap. The CLI version does not provide the second feature. Currently, all MAC addresses are read from a file and then assigned to computers based on the ordering from the file. All other features related to network configuration of cluster nodes are the same.

The Configurator step is the only step that has not been translated to the new interface. Initial work on the Configurator halted when issues with generating the dynamic output and reading back input from the user arose involving the way the Configurator was being stored. These issues are being resolved and hopefully in a future version of OSCAR, the command line Configurator will be completed.

The CLI has two main modes of execution: (i) interactive, and (ii) non-interactive. In interactive mode, the user is presented with menus similar to those of the GUI. To start OSCAR in the interactive CLI mode, the command line is similar to the standard invocation with the addition of one option, e.g., `install_wizard --cli ethX`. The user is led through a series of menus where they select from available options, provide input, and continue from step-to-step in the installation process. The other form enables the user to automate the entire process, avoiding any input from the terminal. In non-interactive mode, more flags can be added to the command line to make the CLI read input from files rather than interactively prompting the user. The command line options for this fully automated (non-interactive) CLI mode are shown in Listing 1.

The node startup and installation phase function differently when working in either interactive or non-interactive mode. In the interactive version of the CLI, the installation pauses at that point and waits for the user to enter some text, but the non-interactive version requires a more automatic method for building nodes. To aid this node automation, there is a flag in the OSCAR CLI installation called `--bootscript` which accepts a user provided script that will be executed to control automation of node boot/build. This script can be as simple as beep and prompt for input from the terminal (interactive for this single step), or use of other tools for remote control of nodes, e.g., remote power controllers, serial consoles, IPMI. Additionally, this scripting hook could be employed to boot virtual machines if working in a virtualized environment, which is common during development and testing phases. The script follows standard UNIX semantics and returns zero (0) on success and non-zero in all other cases.

Listing 1. Flags to automate installation

```
--opkgselector file     #OSCAR Packages Selector

--buildimage file       #Build Client Image

--defineclients file    #Define OSCAR Clients

--networkclients file   #Setup Client Networking

--bootscript file       #Returns 0 when all the
                        #clients have booted with
                        #their new disk images.
```

Note, a hybrid approach can be used where portions of the installation are fully automated and others are interactive. Therefore, any options that are not provided via CLI flags will be acquired through the interactive user input menus. For example, if only one step has a file assigned to it, that step will be the only one that will run without user input. All others steps will run as described in Section 4 and prompt the user for the input required to finish the OSCAR installation. This allows the user to quickly skip past steps that are the same every time and move on to step that require interaction and new input each time.

6 Future Work

While some work on the CLI is completed, there is still much more to do. The issue that currently holds the CLI back the most is the separation of the logic and interface code. From the beginning of the OSCAR project, the graphical interface has been integrated into the logic code in many places, specifically the MAC address selection and the Configurator. Because of these problems, the CLI does not completely mirror the graphical version. Once the separation between the two parts of the OSCAR code takes place, the CLI and the GUI can use the same logic code. Other possibilities would include the GUI using the CLI as back-end code. By doing these two things, the code would be much easier to maintain because there would only be one place that would require updating.

7 Conclusion

This paper presents the design and implementation of a menu based command line interface for OSCAR. This approach provides a powerful solution for both users and developers.

For users, the menu based CLI is a suitable solution for reproducible installations, using the capability of logs from a previous invocations. Also, a command line interface is typically the preferred solutions for remote uses. Since a GUI requires the execution of a graphical environment, which is expensive in term of resource consumption, it is cumbersome for OSCAR users to utilize a graphical wizard interface via remote access methods.

For developers, a menu based CLI provides a major capability for enabling the creation of tools based on OSCAR; the integration of the CLI being simpler than the integration of a graphical windowing environment. For instance, a tool for automatic testing based on the CLI is currently under development. Such a tool is mandatory to improve the quality control procedures that are critical for OSCAR developers since OSCAR currently supports several Linux distributions and hardware architectures, which ultimately leads to complex and expensive testing phases.

References

[1] Benoît des Ligneris and F. L. Camargos. OSCAR CLI: A Command Line Interface for OSCAR. In *Proceeding of* 2^{nd} *Annual OSCAR Symposium (OSCAR 2004)*, Winnipeg, Manitoba Canada, May 16-19 2004.

[2] S. Dague. System Installation Suite Massive Installation for Linux. In *The* 4^{th} *Annual Ottawa Linux Symposium (OLS'02)*, Ottawa, Canada, June 26-29, 2002.

[3] Env-Switcher, http://env-switcher.sourceforge.net.

[4] J. L. Furlani and P. W. Osel. Abstract Yourself With Modules. In *Proceedings of the 10th Large Installation Systems Administration Conference (LISA'96)*, pages 193–204, Chicago, IL, September 29 – October 4, 1996.

[5] B. Luethke, T. Naughton, and S. L. Scott. C3 Power Tools: The Next Generations... In *DAPSYS 2002*, pages 82–89, Johannes Kepler University, September 29 – October 2, 2002. Kluwer Academic Publishers.

[6] J. Mugler, T. Naughton, S. L. Scott, B. Barrett, A. Lumsdaine, J. M. Squyres, Benot des Ligneris, Francis Giraldeau, and C. Leangsuksun. OSCAR Clusters. In *Proceedings of the* 5^{th} *Annual Ottawa Linux Symposium (OLS'03)*, Ottawa, Canada, July 23-26, 2003.

[7] J. Spolsky. *User Interface Design for Programmers*. Apress, 2001.

[8] J. M. Squyres. Meta menu: A state-machine approach to making menu-based systems. Meta Menu Project Web Page, http://sourceforge.net/projects/metamenu/.